The caBIG Proteomics/AML project

Patrick McConnell
Simon Lin
Duke Comprehensive Cancer Center
Duke Bioinformatics Shared Resource

Document version history

Version	Date	Editor	Description
1.0	8-18-04	Simon Lin	General outline
1.1	9-30-2004	Patrick McConnell	First draft
2.0	12-15-2004	Patrick McConnell	Major revision
2.1	12-17-2004	Patrick McConnell	Minor revision
2.2	12-23-2004	Patrick McConnell	Minor revision
2.3	12-29-2004	Patrick McConnell	Minor revision

Table of Contents

Executive Summary

We discuss the need for modeling statistical operations. To satisfy these needs, we propose the Analysis Modeling Language. AML has three core models: the description of an analysis routine, the run of an analysis routine, and a generic model to encode statistical data.

Target audience: data architect

Introduction

Existing standards

MathML

The Math Markup Language is an XML language for describing mathematical equations. This standard does not focus on metadata, but instead describes the actual equation itself, from variables to operations.
http://www.w3.org/Math/

PMML

The Predictive Model Markup Language is an XML-based language which provides a way for applications to define statistical and data mining models and to share models between PMML compliant applications. This language also does not focus on metadata.
http://www.dmg.org/

StatDataML

StatDataML is an XML encoding for statistical data. Parsers exist for R, MATLAB, and Octave. The XML is overly verbose, and there is no support for Base64 encoding.
http://cran.r-project.org/src/contrib/Descriptions/StatDataML.html

ADaM

The Analysis Dataset Model
http://www.cdisc.org/models/adam/V1.0/index.html

Terminology

Markup language: a way to describe something, usually in XML.

Analysis: any algorithm or routine that takes a set of inputs, performs a manipulation on those inputs, and provides outputs.

AML: the analysis markup language, the topic of this paper, is simply a way to describe an algorithm or routine.

Our approach

Modeling analyses takes three major parts:

- Describe an analysis routine
- Describe a run of an analysis routine
- Encoding statistical data

Modeling is not a complete solution to the problem. A controlled vocabulary, or ontology, is also needed to help us interpret the data.

Describing an analysis routine: AML-routine

It is necessary to describe an analysis routine in as much detail as is possible or makes sense. Analysis is the underpinnings of science; so, if an analysis routine is not understood, then it will not be used. Here we describe some of the necessary annotation for an analysis routine.

Authors

The authors of the routine should be included to give them credit for the work as well as provide a contact for questions about the routine. Important fields here include the name of the author, the institution or company the author developed the routine in/for, the role the author took in the development of the routine (e.g. principle investigator, developer, data modeler, etc.), and contact information.

Routine name

Every analysis routine must have a one-word name, bayesianDecomposition for example. This often corresponds to the function or subroutine name in a programming language.

Ontological description

The routine should have a list of controlled vocabulary terms that describe the functionality of the analysis routine. This is to provide semantic information for machine processing.

Textual description

A human-readable textual description of the routine's functionality is necessary so that users of that routine can get a feeling for what it does. This description should include information about expected input and output data, the algorithms employed, and any other textual information that would aid in the understanding of the analysis routine. Typically, the textual description will be on the order of 2-5 sentences.

External references

Any existing external resources, such as web sites or journal articles, should be included to aid in the understanding and application of the analysis routine.

Pseudo-code

Pseudo-code is often used to provide a layman's description of what an algorithm is actually doing. This pseudo code should provide a direct representation of what the algorithm is doing, but not with the many details that full source code employs. For example, text descriptions can replace blocks of meticulous code.

Source code

The ultimate description of an analysis routine is the actual source code for the routine. It is the most descriptive explanation of any algorithm. The source code should be included, if available.

Routine signature

The signature of an analysis routine describes the inputs and outputs of the routine. This does not take the place of a formal language for describing the signature, such as the Web Services Description Language (WSDL), but it does provide some informatics to help the use of the analysis routine to better understand how it is to be invoked. The signature includes a set of input and output parameters, which include the name of the parameter, the order the parameter is to appear in relation to the other parameters, whether the parameter is required, the data type of the parameter, a reference to the WSDL parameter, and finally a textual description of the parameter. The type need not be a formal definition, such as an XML Schema, but is intended instead to provide some information about the type, for example integer or gene.

Contract

The formal contract of the analysis routine provides pre-conditions and post-conditions to invoking the analysis routine. This contract should be specified in a formal language, such as OCL or Java.

Implementation

The implementation of an analysis routine describes how it is being run. The important information includes the platform (such as x86 or SGI), operating system (such as Windows XP Professional or Irix 6.5), and the compiler used to generate the machine code or the interpreter/virtual machine used to run the code.

Caveats

A caveat is some piece of information or discrepancy that the user may be interested in knowing but is not covered under any of the other fields. An example would be that a particular input produces spurious results on a given platform.

Benchmarks

Benchmarks describe the performance of an analysis routine. This performance is represented by Big-O notation, describing the amount of work that must be done in relation to the size of the input. In addition, references to actual runs that model this performance should be included.

Describing an analysis run: AML-run

It is necessary to describe how an analysis routine is invoked: the input data, the output data. The underpinnings of experimental science is repeatability – if an experiment is not repeatable, it is not valid. In the same way, if a statistical analysis is

not repeatable, it is not valid. Thus, we need a way to describe the invocation of an analytical routine such that it is easily repeatable.

From the perspective of a data architect, an analysis element is the glue between data and describes how data is generated. From the perspective of a scientist or statistician, an analysis element describes how to reproduce a particular statistical experiment.

Analysis runs can be organized into sets, which can represent an analytical workflow, different runs on the same analysis routine but with different parameters, or any other useful organization.

Here, we describe some of the data that is necessary to describe an analysis run:

Analysis routine

This is the analysis routine that is invoked, which takes the form of a pointer to the aml-routine document (see previous section) as well as the service descriptor used to invoke the service.

Invoker

The invoker is a description who submitted the request to the routine, as well as the time the submittal took place and the time the analysis routine completed.

Input and output data

In addition to tracking which analysis routine was invoked by whom, the final crucial piece of information is the input and output data. This takes the form of a list of data pointers that point to the piece of data passed to the analysis routine. Additionally, the name of the parameter is associated with each data pointer.

User comments

Each analysis run can have one or more user-entered descriptions of that run. These are anecdotal annotations of that particular run. Examples can include the success of the run and comments on the parameters used.

Encoding statistical data: StatML

Translating data between software systems and programming languages can be very tricky and take a large portion of software development time. This is especially true of statistical data. Much time is spent by a statistician in the parsing and serializing of data. Furthermore, there are some languages that are more suited to certain tasks than others. For example, Java has good library support for graphics, network interaction, and parsing, whereas R as good library support for a variety of statistical operations. We seek to bridge this gap by providing a generic encoding for statistical data.

Lists

Lists are generic collections of lists, arrays, and scalars. They must contain the length of the list, but may optionally be named and, in the case of homogenous lists, have a data type.

List context

The first element of a list may be a set of attributes, which is a list of named elements that describe the list. These attributes are important for describing "metadata" associated with the list. Examples include the row and column names, whether the list is a data frame, and the multiple names that may be associated with the list.

Arrays

An array is a set of scalar values, organized into blocks that correspond to dimensions. For example, an array with dimensions "2,3" is a rectangular array, two columns of three elements. In addition to dimensions, arrays also can be named and have a data type. Array values are stored in their binary format, base64 encoded. Base64 encoding takes three bytes of data and converts it to four characters. Double values use eight bytes and integer values use four bytes. The encoding of these bytes must be big-endian, which means that the most significant bit is in the lower bytes (most left).

Scalars

A scalar is a single value with data type double, integer, string, or boolean. A scalar has an optional name.

Ontology

An ontology is a controlled vocabulary used to add semantic information to data. Semantic information helps us understand terminology, provide enumerations of possible data values, and provides relationships between terms. This metadata helps both humans and machines better interpret data. All fields in markup language should be described semantically. However, there are also some fields that can be enumerated with a controlled vocabulary. We describe some of these fields here.

AML-routine

The most obvious place for ontology in the AML-routine is the ontology section, where controlled vocabulary for describing the routine is placed. Other useful fields to be described by ontology include source code language, data type for parameters, and the platforms/operating systems/compilers for the implementation.

AML-run

AML-run is simply the glue that connects data and the AML-routine invoked on the data, so most of the ontology is found in those data definitions, and there is little place for semantics in AML-run outside of field descriptions.

StatML

In StatML, the most prominent enumerated field is the data type field. The list attributes field is another place that ontology could be employed.

Examples

See files: aml-routine-example.xml, aml-run-example.xml and statml-example.xml.